

O constanta de tip sir de caractere se declara intre doua caractere “. In memoria interna, o constanta de acest tip este retinuta sub forma unui vector de caractere. Fiecare componenta a sirului (incepand cu cea de indice 0) retine codul ASCII al caracterului pe care il memoreaza. Conventia este ca ultimul octet sa retina 0 (codul caracterului nul). Caracterul nul este memorat automat. Trebuie rezervate *lungimea_sirului*+1 caractere char (+1 pentru caracterul nul).

Limbajul C/C++ permite initializarea unui tablou de caractere printr-o constanta sir, care include automat caracterul null.

Exemplu :

```
char vect[11]="calculator";
char vect[]="calculator"; (compilatorul face calculul numarului de octeti necesari)
char vect[100]="calculator"; (s-au rezervat mai multi octeti decat era necesar)
```

Sirurile de caractere sunt de fapt *tablouri de caractere*, care au ca ultim element un terminator de sir, caracterul null.

Exemplu:

```
char tc[5] = {'a', 'b', 'c', 'd', 'e'}; // tablou de caractere
char sc[5] = {'a', 'b', 'c', 'd', '\0'}; // sir de caractere cu elementele abcd
```

Ultima initializare este echivalenta cu:

```
char sc[5] = "abcd"; //sau char sc[] = "abcd";
char sc1[5] = "abcd";
char s[10];
cout<<sc<<endl; //afiseaza abcd
cout<<tc<<endl; //eroare: tabloul de caractere nu contine terminatorul de sir, deci nu poate fi afisat
ca sir
cout<<s<<endl; // eroare: tablou neinitializat
cout<<sc1[0]; // afiseaza primul caracter din sirul sc1
cout<<sc1[2]; // afiseaza al treilea element din sirul sc1
sc1[1]='K'; // elementului din sir de indice 1 i se atribuie valoarea 'K';
```

CITIREA / AFISAREA SIRURILOR DE CARACTERE

Sirurile de caractere pot fi initializate inca de la declarare sau citite pe parcursul programului.

a. Citirea unui sir de caractere se poate face ca citirea oricarui tablou, intr-un for, caracter cu caracter (desi nu este recomandata). In acest caz, terminatorul de sir nu este memorat automat, el trebuie pus explicit dupa ultimul caracter din sir.

Exemplu:

```
char c[20];
for(int i=0;i<=5;i++)
    cin>>c[i];
cout<<c<<endl; //se va afisa sirul format din cele 6 caractere, urmat de caractere „reziduale”,
//initializate implicit la compilare, din cauza ca n-a fost pus terminatorul de sir
c[6]=0;
cout<<c<<endl; //a fost pus terminatorul de sir, deci sirul va fi afisat corect
```

b. Se poate face pur si simplu, folosind *cin>>*. Caracterul nul este adaugat automat. Dezavantajul este ca in acest fel nu se pot citi siruri care contin mai multe cuvinte separate prin spatii. Citirea sirului se sfarseste la intalnirea primului caracter blank (de ex, daca se citeste “ora de informatica”, variabila c va retine numai “ora”).

Exemplu

```
char c[30];
cin>>c;
cout<<c;
```

c. Se poate folosi o functie speciala pentru citirea sirurilor de caractere, inclusa in biblioteca *string.h* (varianta recomandata).

Exemplu

```
char a[30],x;int nr;
cin.get(a,nr,x);
```

Funcția **cin.get** citește un sir de caractere sau până când au fost citite nr-1 caractere, sau dacă s-a întâlnit caracterul x. Al treilea parametru poate lipsi, caz în care el este implicit caracterul '\n' (new line). Sunt citite și caracterele albe, caracterul nul este inserat automat iar caracterul transmis ca ultim parametru nu este inserat în sir.

Exemplu

```
char a[30];
cin.get(a,5,'s');           //dacă se citește sirul "maimuta, variabila a va reține "maim"
cin.get(a,15,'s');          //dacă se citește sirul "maimuta, variabila a va reține "maimuta"
cin.get(a,15,'t');          //dacă se citește sirul "maimuta, variabila a va reține "maimu"
cin.get(a,4,'t');           //dacă se citește sirul "maimuta, variabila a va reține "mai"
cin.get(a,10);              //dacă se citește sirul "maimuta, variabila a va reține "maimuta"
```

Funcția **cin.get()** fără parametri are rolul de a citi un caracter (alb sau nu).

Funcția **cin.get(char c)** are rolul de a citi un caracter (alb sau nu) pe care îl încarcă în variabila c.

Observație: În cazul utilizării repetate a funcției **cin.get(a,nr,x)**, după fiecare folosire trebuie citit caracterul de la sfârșitul fiecărui sir, adică '\n' (în caz contrar, acest caracter va fi încărcat la începutul următorului sir, a cărui citire se termină la caracterul Enter, deci citirea celui de-al doilea sir se termină înainte de a începe, iar al doilea sir va fi sirul vid). Aceasta citire a caracterului '\n' se realizează folosind **cin.get()** fără parametri.

Exemplu

```
char a[30],b[30];
cin.get(a,15);
cin.get(b,10);
```

Dacă se încearcă citirea sirurilor „sarbatoare” și „vacanta”, se observă că a=”sarbatoare”, b=”” (nici nu apucăm să citim sirul b). Varianta corectă este:

```
cin.get(a,15);
cin.get();
cin.get(b,10);
```

Afișarea unui sir de caractere se face folosind **cout**.

```
cout<<a;
```

Se poate afișa și caracter cu caracter, ca în cazul tablourilor, dar această variantă nu este recomandată.

FUNCTII PENTRU OPERATII CU SIRURI DE CARACTERE

Funcțiile pentru operații cu siruri se găsesc în header-ul **<string.h>**.

➤ Funcția strlen

int strlen(nume_sir); – returnează lungimea efectivă a unui sir (fără a număra terminatorul de sir).

Exemplu:

```
char a[50]="ora de informatica"; → strlen(a) = 18
```

➤ Funcția strcpy

strcpy(sir_destinatie, sir_sursa); – copiază sirul sir_sursa în sir_destinatie (se simulează atribuirea a=b).

ATENȚIE!! Nu este permisă atribuirea între două siruri de caractere folosind operatorul =. Atribuirea se face folosind funcția **strcpy**.

Exemplu:

```
char a[50]="primul sir",b[40]="al doilea sir";
```

```
a=b; //eroare
strcpy(a,b); → a = "al doilea sir"; b="al doilea sir";
```

➤ Functia **strcat**

`strcat(dest, sursa);` – adauga sirului `dest` sirul `sursa`. Sirul `sursa` ramane nemodificat. Operatia se numeste *concatenare* si nu este comutativa.

Exemplu:

```
char *a="vine ", *b="vacanta?";
strcat(a,b); → a = "vine vacanta?";
```

➤ Functia **strncat**

`strncat(dest, sursa, nr);` – adauga `dest` primele `nr` caractere din sirul `sursa`. Sirul `sursa` ramane nemodificat.

Exemplu:

```
char *a="vine ", *b="vacanta?";
strncat(a,b,4); → a = "vine vaca";
```

➤ Functia **strchr**

`strchr(sir, c);` – are rolul de a cauta caracterul `c` in sirul `sir`. Cautarea se face de la stanga la dreapta, iar functia intoarce adresa subsirului care incepe cu prima aparitie a caracterului `c`. Daca nu este gasit caracterul, functia returneaza 0. Diferenta dintre adresa sirului initial si cea a subsirului returnat reprezinta chiar pozitia caracterului cautat in sirul dat.

Exemplu:

```
char *a="acesta este un sir", b='t', c='x', d;
cout<<strchr(a,b); → se tipareste "ta este un sir";
cout<<strchr(a,c); → nu se tipareste nimic (se tipareste 0 daca se face o conversie la int a
lui strchr(a,c) ;
d= strchr(a,b);
cout<<"Caracterul apare prima data la pozitia "<<d-a;
```

Ex: Sa se afiseze toate pozitiile unui caracter intr-un sir

```
#include <iostream.h>
#include <string.h>
void main()
{char a[100], *p, c;
cin.get(a, 100);
cin>>c;
p=strchr(a, c);
while (p)
{cout<<"Pozitia "<<p-a<<endl;
p++;
p=strchr(p, c);}}
```

➤ Functia **strrchr**

`strrchr(sir, c);` – are acelasi rol cu `strchr`, cu deosebirea ca returneaza adresa ultimei aparitii a caracterului (cautarea se face de la dreapta spre stanga; `r = right`)

➤ Functia **strcmp**

`int strcmp(sir1, sir2);` – are rolul de a compara doua siruri de caractere. Valoarea returnata este `<0` (daca `sir1<sir2`), `=0` (daca `sir1=sir2`) si `>0` (daca `sir1>sir2`). Functia `strcmp` face distinctie intre literele mari si cele mici ale alfabetului.

Obs: Functia **strcmp** returneaza diferenta dintre codurile ASCII ale primelor caractere care nu coincid

➤ Functia **stricmp**

`int stricmp(sir1, sir2);` – are acelasi rol cu `strcmp`, cu deosebirea ca nu face distinctie intre literele mari si cele mici ale alfabetului (i = ignore).

➤ Functia **strstr**

`strstr(sir1, sir2);` – are rolul de a identifica daca sirul `sir2` este subsir al sirului `sir1`. Daca este, functia returneaza adresa de inceput a subsirului `sir2` in sirul `sir1`, altfel returneaza adresa 0. In cazul in care `sir2` apare de mai multe ori in `sir1`, se returneaza adresa de inceput a primei aparitii. Cautarea se face de la stanga la dreapta

➤ Functia **strtok**

`strtok(sir1, sir2);` – are rolul de a separa sirul `sir1` in mai multe siruri (cuvinte) separate intre ele prin unul sau mai multe caractere cu rol de separator. Sirul `sir2` este alcatuit din unul sau mai multe caractere cu rol de separator.

Functia **strtok** actioneaza in felul urmator:

- Primul apel trebuie sa fie de forma `strtok(sir1, sir2)`; Functia intoarce adresa primului caracter al primei entitati. Dupa prima entitate, separatorul este inlocuit automat prin caracterul nul.
- Urmatoarele apeluri sunt de forma `strtok(NULL, sir2)`; De fiecare data, functia intoarce adresa de inceput a urmatoarei entitati, adaugand automat dupa ea caracterul nul.
- Cand sirul nu mai contine entitati, functia returneaza adresa nula.

Exemplu:

```
//Sa se separe cuvintele dintr-un text.
#include <iostream.h>
#include <conio.h>
#include <string.h>
void main()
{char text[100], cuv[10][10], *p, *r, separator[]=", . !?"; int i=0, nr=0;
clrscr();
cout<<"Dati sirul:"; cin.get(text, 100);
strcpy(p, text);
p=strtok(p, separator);
while (p)
{strcpy(cuv[++nr], p);
p=strtok(NULL, separator);}
cout<<"Sunt "<<nr<<" cuvinte:"<<endl;
for (i=1; i<=nr; i++) cout<<cuv[i]<<endl;
getch();}
```

➤ Functia **strspn** cu forma generala

`int strspn(sir1, sir2);` – are rolul de a returna numarul de caractere ale sirului `sir1` (caractere consecutive care incep obligatoriu cu primul caracter) care se gasesc in sirul `sir2`.

Exemplu:

`strspn("AB2def", "1B3AQW");` → returneaza 2, pentru ca primele 2 caractere 'A' si 'B' din `sir1` se gasesc in `sir2`.
`strspn("FAB2def", "16A32BF");` → returneaza 0, deoarece caracterul 'F' cu care incepe `sir1` nu se gaseste in `sir2`.

➤ Functia **strcspn** cu forma generala

`int strcspn(sir1, sir2);` – are rolul de a returna numarul de caractere ale sirului `sir1` (caractere consecutive care incep obligatoriu cu primul caracter) care nu se gasesc in sirul `sir2`.

Exemplu:

`strspn("AB2def","123");` → returneaza 2, pentru ca primele 2 caractere din `sir1` nu se gasesc in `sir2`.

//Se citeste un sir de caractere care nu contine caractere albe. Sa se decida daca sirul este alcatuit exclusiv din caractere numerice.

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
void main()
{char text[100],cifre[]="0123456789";
clrscr();
cout<<"Dati sirul:";cin.get(text,100);
if (strcspn(cifre,text)==strlen(text))
cout<<"exclusiv numeric";
else cout<<"nenumeric";
getch();}
```

➤ Functia **strlwr** cu forma generala

`strlwr(sir);` – are rolul de a converti toate literele mari din `sir` in litere mici. Restul caracterelor raman neschimbate.

➤ Functia **strupr** cu forma generala

`strupr(sir);` – are rolul de a converti toate literele mici din `sir` in litere mari. Restul caracterelor raman neschimbate

➤ Functia **strbrk** cu forma generala

`strpbrk(sir1,sir2);` – actioneaza in felul urmator:

- Cauta primul caracter al sirului `sir1` in `sir2`. Daca este gasit, returneaza adresa sa din cadrul sirului `sir1` si executia se termina. Altfel, se trece la pasul urmator.
- Cauta al doilea caracter al sirului `sir1` in `sir2`. Daca este gasit, returneaza adresa sa din cadrul sirului `sir1` si executia se termina. Altfel, se trece la pasul urmator.
- ...
- Daca nici un caracter al sirului `sir1` nu apartine sirului `sir2`, functia returneaza adresa nula.

➤ Functia **atof** cu forma generala

`double atof(sir);` – converteste un sir catre tipul `double`. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0. Aceasta functie (ca si cele similare) necesita includerea libreriei `stdlib.h`.

➤ Functia **_atold** cu forma generala

`long double _atold(sir);` – converteste un sir catre tipul `long double`. Daca aceasta conversie esueaza, valoarea intoarsa este 0.

➤ Functia **atoi** cu forma generala

`int atoi(sir);` – converteste un sir catre tipul `int`. Daca aceasta conversie esueaza (se intalneste un caracter nenumeric), valoarea intoarsa este 0.

➤ Functia **atol** cu forma generala

`long atol(sir);` – convertește un sir către tipul `long`. Dacă această conversie esuează (se întâlnește un caracter nenumeric), valoarea întoarsă este 0.

➤ Functia **itoa** cu forma generala

`itoa(int valoare, sir, int baza);` – convertește o valoare de tip `int` în sir, care este memorat în variabila `sir`. Baza reține baza de numeratie către care să se facă conversia. În cazul bazei 10, sirul reține și eventualul semn `-`.

➤ Functia **ltoa** cu forma generala

`ltoa(long valoare, sir, int baza);` – convertește o valoare de tip `long int` în sir, care este memorat în variabila `sir`.

➤ Functia **ultoa** cu forma generala

`ultoa(unsigned long valoare, sir, int baza);` – convertește o valoare de tip `unsigned long` în sir, care este memorat în variabila `sir`.

Probleme propuse:

1. Să se verifice dacă un cuvânt este palindrom.
2. Să se transforme un sir din litere mici în litere mari.
3. Să se transforme un sir din litere mari în litere mici.
4. Să se determine frecvența de apariție a unui caracter într-un text.
5. Să se genereze toate prefixele / sufixele unui cuvânt.
6. Se citește un text dintr-un fișier și un caracter `c`. Să se determine de câte ori se găsește caracterul în text (nu se face distincție între literele mari și literele mici).
7. Se citește un text de la tastatură astfel încât cuvintele să fie separate printr-un singur spațiu și imediat după ultimul cuvânt se scrie punct. Textul va fi scris pe un singur rând.
 - a) Să se determine câte cuvinte conține textul. De ex : "Ana are mere." Conține 3 cuvinte.
 - b) Să se determine dacă textul are cuvinte distincte (se ignoră diferența de cheie).
 - c) Să se determine dacă textul conține cifre.
8. Să se determine de câte ori se găsește un cuvânt într-un text.
9. Codificați un text astfel încât litera `a` să devină `c`, `b` să devină `e` s.a.m.d.
10. Să se sorteze alfabetic un sir de cuvinte (eventual, fără a distinge literele mici de cele mari).
11. Codificarea pasarească a unui cuvânt (după fiecare vocală, se pune litera `p` urmată de aceea vocală). Ex : `informatica` → `ipinfopormapatipicapa`
12. Se citesc `n` cuvinte. Să se afișeze grupurile de cuvinte care rimează (au ultimele 2 caractere identice).
13. Să se despartă un text în cuvinte și să se afișeze cuvintele separate. Să se afișeze cuvântul de lungime maximă.
14. Să se verifice dacă două cuvinte sunt sau nu anagrame.
15. Să se numere aparițiile unui cuvânt într-un text.
16. Se citește un număr real de la tastatură. Să se trunchieze astfel încât cifrele rămase să formeze o secvență monotona.
Ex. `34.59483` → `34.59` ; `24.1276` → `24`
17. Se citește un sir de caractere alfanumerice. Considerăm ca literele sunt separatorii numerelor. Afișați datele de tip numeric preluate în ordine din sirul citit. Numerele vor fi scrise câte unul pe o linie.

Ex.

<i>in.txt</i>	<i>out.txt</i>
<code>a23sc345ss5e</code>	<code>23</code>
	<code>345</code>
	<code>5</code>

18. In directorul curent se afla fisierul cuvinte.txt care contine mai multe linii de text formate din cuvinte separate de cate un spatiu. Sa se afiseze cuvintele care au cel putin 3 consoane sau 3 vocale consecutive.
19. Se citeste un sir de caractere. Sa se afiseze sirul oglindit din care lipsesc vocalele.
20. Se da un text de maxim 30 de caractere. Sa se listeze toate cuvintele de doua caractere din acest text.
21. Se introduc de la tastatura cuvinte pana la citirea cuvantului stop. Afisati pe ecran cuvintele mai mari in sens lexicografic decat primul citit.
22. Se dau doua texte. Sa se stabileasca o vocala comuna celor doua texte, care apare de cele mai putine ori.
23. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se determine cate cuvinte contine textul.
24. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Se va genera un nou text care va contine cuvintele ordonate alfabetic
25. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se scrie intr-un alt fisier, pe linii separate, fiecare cuvant care apare in text urmat de un numar care va reprezenta de cate ori apare cuvantul in text. Sa se determine cuvantul care apare de cele mai multe ori.
26. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Intr-un alt fisier se va scrie pe linii separate fiecare cuvant si liniile pe care apare.
27. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu sau mai multe. Se va genera un nou fisier care va contine textul initial avand spatiile de prisos eliminate (intre cuvinte va ramane numai cate un spatiu).
28. Se citesc de la tastatura elementele unei matrici de caractere (nr. linii=nr. coloane), $A(N \times N)$, $N \leq 10$.
 - a) Sa se afiseze matricea A ;
 - b) Sa se formeze si sa se afiseze cuvantul format din caracterele pe pe diagonala principala a matricii A ;
 - c) Sa se calculeze si sa se afiseze numarul de litere mari, litere mici si cifre din matrice;
 - d) Sa se afiseze cuvantul format din caracterele de pe diagonala secundara;
 - e) Sa se afiseze procentul literelor mari, al literelor mici si al cifrelor de pe cele 2 diagonale;
 - f) Sa se afiseze caracterele comune aflate pe liniile p si q ($p, q < N$, p si q citite de la tastatura);
 - g) Sa se afiseze in ordine alfabetica, crescatoare, literele mari aflate pe coloanele impare.
29. Simulati comanda REPLACE astfel incat intr-un text veti inlocui un caracter x citit de la tastatura cu un alt caracter y citit de la tastatura. Se ignora sau nu diferenta de cheie dupa optiunea utilizatorului.
30. Simulati comanda REPLACE astfel incat intr-un text veti inlocui un sir x citit de la tastatura cu un alt caracter sir y citit de la tastatura. Se ignora sau nu diferenta de cheie dupa optiunea utilizatorului.
31. Se citeste de la tastatura un cuvant. Sa se stabileasca daca el contine doua litere alaturate identice, afisandu-se un mesaj corespunzator.
32. Dintr-un fisier se citesc numele a n persoane. Sa se modifice continutul fisierului astfel incat toate numele sa fie scrise astfel: prima litera mare si restul litere mici.
33. Se citesc n siruri. Pentru fiecare sir se va determina suma codurilor ASCII.
34. Intr-un fisier sunt scrise cuvinte pe linii separate. Sa se afiseze cuvintele care contin majuscule.
35. Intr-un fisier sunt scrise pe randuri diferite numele a n copii. Sa se modifice continutul fisierului astfel incat sa contina numele ordonate crescator.
36. Sa se afiseze vocalele unui cuvant.
37. Sa se afiseze cuvintele care incep si se termina cu consoana, (vocala) etc.
38. Sa se desparta un text in cuvinte si sa se afiseze cuvintele separate. Sa se afiseze cuvantul de lungime maxima.
39. Intr-un text exista un cuvant. Codificati/decodificati cuvantul dupa un algoritm generat de voi.
40. Aceeasi problema pentru un text.
41. Se dau doua texte. Sa se stabileasca o vocala comuna celor doua texte, care apare de cele mai putine ori.

42. Dintr-un fisier se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se determine cate cuvinte contine textul.
43. Se citesc n cuvinte. Sa se afiseze perechile de 2 cuvinte care rimeaza.
44. Aceeasi problema, numai ca se vor afisa grupurile de cuvinte care rimeaza
45. Se citesc cuvinte pana la citirea cuvintului "stop". Sa se afiseze cate dintre cuvintele citite sunt egale cu primul cuvant citit.
46. Se citeste un text. Textul contine cuvinte separate printr-un spatiu. Se va genera un nou text care va contine cuvintele ordonate alfabetic
47. Se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se scrie, pe linii separate, fiecare cuvant care apare in text urmat de un numar care va reprezenta de cate ori apare cuvantul in text. Sa se determine cuvantul care apare de cele mai multe ori.
48. Se citeste un text. Textul contine cuvinte separate printr-un spatiu. Sa se scrie, pe linii separate, fiecare cuvant si liniile pe care apare.
49. Se citeste un text care contine cuvinte separate printr-un spatiu sau mai multe. Se va genera un nou text care va contine textul initial avand spatiile de prisos eliminate (intre cuvinte va ramane numai cate un spatiu).
50. Simulati scrierea unei parole intr-un fisier. La tastarea parolei pe ecran in locul fiecarui caracter se va scrie caracterul '*'. Eventual realizati si incryptarea parolei inainte de a fi scrisa intr-un fisier.
51. Fie un sir de forma: cifra-litera, cifra litera ...etc. Ex 2a4b5c. Sa se genereze un astfel de sir: aabbbbcccc.
52. Fie un sir format din replicarea de un numar de ori a unui subsir redundant. Sa se determine cea mai scurta secventa redundantă. Se va afisa subsirul redundant si numarul sau de aparitii. Ex pt: aabcaabcaabcaabc se va fisa: aabc si nu aabcaabc. Numarul de aparitii este 4 si nu 2.